Basic R Programming: Exercises

R Programming
John Fox

ICPSR, Summer 2009

1. *Logistic Regression:* Iterated weighted least squares (IWLS) is a standard method of fitting generalized linear models to data. As described in Section 5.5 of *An R and S-PLUS Companion to Applied Regression* (Fox, 2002), the IWLS algorithm applied to binomial logistic regression proceeds as follows:

   (a) Set the regression coefficients to initial values, such as $\boldsymbol{\beta}^{(0)} = \mathbf{0}$ (where the superscript 0 indicates start values).

   (b) At each iteration t calculate the current fitted probabilities $\boldsymbol{\mu}$, variance-function values $\boldsymbol{\nu}$, working-response values $\mathbf{z}$, and weights $\mathbf{w}$:

   $$\begin{aligned} \mu_i^{(t)} &= [1 + \exp(-\eta_i^{(t)})]^{-1} \\ v_i^{(t)} &= \mu_i^{(t)}(1 - \mu_i^{(t)}) \\ z_i^{(t)} &= \eta_i^{(t)} + (y_i - \mu_i^{(t)})/v_i^{(t)} \\ w_i^{(t)} &= n_i v_i \end{aligned}$$

   Here, n_i represents the binomial denominator for the ith observation; for binary data, all of the n_i are 1.

   (c) Regress the working response on the predictors by weighted least squares, minimizing the weighted residual sum of squares

   $$\sum_{i=1}^{n} w_i^{(t)}(z_i^{(t)} - \mathbf{x}_i'\boldsymbol{\beta})^2$$

   where $\mathbf{x}_i'$ is the ith row of the model matrix.

   (d) Repeat steps 2 and 3 until the regression coefficients stabilize at the maximum-likelihood estimator $\widehat{\boldsymbol{\beta}}$.

   (e) Calculate the estimated asymptotic covariance matrix of the coefficients as

   $$\widehat{\mathcal{V}}(\widehat{\boldsymbol{\beta}}) = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}$$

   where $\mathbf{W} = \text{diag}\{w_i\}$ is the diagonal matrix of weights from the last iteration and $\mathbf{X}$ is the model matrix.

   Problem: Program this method in R. The function that you define should take (at least) three arguments: The model matrix `X`; the response vector of observed proportions `y`; and

the vector of binomial denominators `n`. I suggest that you let `n` default to a vector of `1`s (i.e., for binary data, where `y` consists of `0`s and `1`s), and that you attach a column of `1`s to the model matrix for the regression constant so that the user does not have to do this when the function is called.

Programming hints:

- Adapt the structure of the example developed in Section 8.5.1 of "Writing Programs" (Fox and Weisberg, draft), but note that this example is for *binary* logistic regression, while the current exercise is to program the more general *binomial* logit model.
- Use the `lsfit` function to get the weighted-least-squares fit, calling the function as `lsfit(X, z, w, intercept=FALSE)`, where `X` is the model matrix; `z` is the current working response; and `w` is the current weight vector. The argument `intercept=FALSE` is needed because the model matrix already has a column of `1`s. The function `lsfit` returns a list, with element `$coef` containing the regression coefficients. See `?lsfit` for details.
- One tricky point is that `lsfit` requires that the weights (`w`) be a *vector*, while your calculation will probably produce a *one-column matrix* of weights. You can coerce the weights to a vector using the function `as.vector`.
- Return a list with the maximum-likelihood estimates of the coefficients, the covariance matrix of the coefficients, and the number of iterations required.
- You can test your function on the `Mroz` data in the **car** package, being careful to make all of the variables numeric. You might also try fitting a binomial (as opposed to binary) logit model.

2. *A Challenging Problem — Ordered Logit and Probit Models:* Ordered logit and probit models are popular regression models for ordinal response variables; the ordered logit model is also called the proportional-odds model (see below for an explanation). The following description is adapted from Fox, *Applied Regression Analysis and Generalized Linear Models, Second Edition* (2008, Ch. 14):

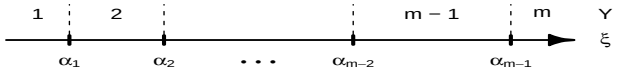
   Imagine that there is a latent (i.e., unobservable) variable ξ that is a linear function of X's plus a random error:

   $$\xi_i = \alpha + \beta_1 X_{i1} + \cdots + \beta_k X_{ik} + \varepsilon_i$$

   The latent response ξ is dissected by $m - 1$ *thresholds* (i.e., boundaries) into m regions. Denoting the thresholds by $\alpha_1 < \alpha_2 < \cdots < \alpha_{m-1}$, and the resulting response by Y, we observe

   $$Y_i = \begin{cases} 1 & \text{if } \xi_i \leq \alpha_1 \\ 2 & \text{if } \alpha_1 < \xi_i \leq \alpha_2 \\ \vdots & \\ m-1 & \text{if } \alpha_{m-2} < \xi_i \leq \alpha_{m-1} \\ m & \text{if } \alpha_{m-1} < \xi_i \end{cases} \tag{1}$$

   The thresholds, regions, and corresponding values of ξ and Y are represented graphically in the following figure. Notice that the thresholds are not in general uniformly spaced.



Using Equation 1, we can determine the cumulative probability distribution of Y:

$$\begin{aligned}\Pr(Y_i \leq j) &= \Pr(\xi_i \leq \alpha_j) \\ &= \Pr(\alpha + \beta_1 X_{i1} + \cdots + \beta_k X_{ik} + \varepsilon_i \leq \alpha_j) \\ &= \Pr(\varepsilon_i \leq \alpha_j - \alpha - \beta_1 X_{i1} - \cdots - \beta_k X_{ik})\end{aligned}$$

If the errors ε_i are independently distributed according to the standard normal distribution, then we obtain the ordered probit model. If the errors follow the similar logistic distribution, then we get the ordered logit model. In the latter event,

$$\begin{aligned}\text{logit}\left[\Pr(Y_i \leq j)\right] &= \log_e \frac{\Pr(Y_i \leq j)}{\Pr(Y_i > j)} \\ &= \alpha_j - \alpha - \beta_1 X_{i1} - \cdots - \beta_k X_{ik}\end{aligned}$$

Equivalently,

$$\begin{aligned}\text{logit}\left[\Pr(Y_i > j)\right] &= \log_e \frac{\Pr(Y_i > j)}{\Pr(Y_i \leq j)} \\ &= (\alpha - \alpha_j) + \beta_1 X_{i1} + \cdots + \beta_k X_{ik}\end{aligned} \tag{2}$$

for $j = 1, 2, \ldots, m-1$.

The logits in Equation 2 are for cumulative categories—at each point contrasting categories above category j with category j and below. The slopes for each of these regression equations are identical; the equations differ only in their intercepts.

Put another way, for a fixed set of X's, any two different cumulative log-odds (i.e., logits)—say, at categories j and j'—differ only by the constant $(\alpha_j - \alpha_{j'})$. The odds, therefore, are *proportional* to one another; that is,

$$\frac{\text{odds}_j}{\text{odds}_{j'}} = \exp\left(\text{logit}_j - \text{logit}_{j'}\right) = \exp(\alpha_j - \alpha_{j'}) = \frac{e^{\alpha_j}}{e^{\alpha_{j'}}}$$

where, for example, $\text{odds}_j = \Pr(Y_i > j)$ and $\text{logit}_j = \text{logit}\left[\Pr(Y_i > j)\right]$. For this reason, Equation 2 is called the *proportional-odds logit model*.

There are $(k+1)+(m-1) = k+m$ parameters to estimate in the proportional-odds model, including the regression coefficients α, $\beta_1, \ldots, \beta_k$ and the category thresholds $\alpha_1, \ldots, \alpha_{m-1}$. Note, however, that there is an extra parameter in the regression equations (Equation 2), because each equation has its own constant, $-\alpha_j$, along with the common constant α. A simple solution is to set $\alpha = 0$ (and to absorb the negative sign into α_j), producing

$$\text{logit}\left[\Pr(Y_i > j)\right] = \alpha_j + \beta_1 X_{i1} + \cdots + \beta_k X_{ik}$$

and thus

$$\begin{aligned}\Pr(Y_i > j) &= \Lambda\left(\alpha_j + \beta_1 X_{i1} + \cdots + \beta_k X_{ik}\right), j = 1, \ldots, m-1 \quad (3)\\ &= \Lambda\left(\alpha_j + \mathbf{x}_i'\boldsymbol{\beta}\right) \quad (4)\end{aligned}$$

where $\Lambda(\cdot)$ is the cumulative logistic distribution. In this parametrization, the intercepts α_j are the *negatives* of the category thresholds. The ordered probit model is similar with

$$\begin{aligned}\Pr(Y_i > j) &= \Phi\left(\alpha_j + \beta_1 X_{i1} + \cdots + \beta_k X_{ik}\right), \ j = 1, \ldots, m-1 \quad (5)\\ &= \Phi\left(\alpha_j + \mathbf{x}_i'\boldsymbol{\beta}\right) \quad (6)\end{aligned}$$

where $\Phi(\cdot)$ is the cumulative normal distribution.

The log-likelihood under both the ordered logit and ordered probit model takes the following form:

$$\log_e L(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{i=1}^{n} \log_e \pi_{i1}^{w_{i1}} \pi_{i2}^{w_{i2}} \cdots \pi_{im}^{w_{im}}$$

where $\boldsymbol{\alpha}$ is an $m-1 \times 1$ vector containing all of the regression constants and $\boldsymbol{\beta}$ is the $k \times 1$ vector containing the other regression coefficients; $\pi_{ij} = \Pr(Y_i = j)$ (i.e., the probability under the model that individual i is in response category j); and the w_{ij} are indicator variables equal to 1 if individual i is observed in category j and 0 otherwise. Thus, for each individual, only one of the w_{ij} is equal to 1 and only the corresponding π_{ij} contributes to the likelihood. Note that for either the ordered logit model or the ordered probit model, the individual-category probabilities can be computed as differences between adjacent cumulative probabilities from Equation 3 or 5 (which are functions of the parameters):

$$\begin{aligned}\pi_{i1} &= 1 - \Pr(Y_i > 1)\\ \pi_{i2} &= \Pr(Y_i > 1) - \Pr(Y_i > 2)\\ &\vdots\\ \pi_{im} &= \Pr(Y_i > m-1)\end{aligned}$$

Problem: Program the ordered-logit model or the ordered-probit model (or both). The function that you define should take (at least) two arguments: The model matrix `X` and the response vector `y`, which should be a factor or ordered factor; I suggest that you attach a column of `1`s to the model matrix for the regression constants so that the user does not have to do this when the function is called; the ordered logit and probit models always have constants. Your function can include an argument to indicate which model — logit or probit — is to be fit.

Programming hints:

- The parameters consist of the intercepts and the other regression coefficients, say `a` and `b`. Although there are cleverer ways to proceed, you can set `b` to a vector of zeroes to start, and compute start values for `a` from the marginal distribution of the response; e.g.,

  ```
  marg.p <- rev(cumsum(rev(table(y)/n)))[-1]
  a <- log(marg.p/(1 - marg.p))
  ```

  Here `y` is the response vector and `n` is the number of observations.

- If you're fitting the ordered logit model, use the cumulative logistic distribution function `plogis()`; if you're fitting the ordered probit model, use the cumulative normal distribution function `pnorm()`.
- Use `optim()` to maximize the likelihood, treating the `lreg2()` function in Section 8.5.1 of "Writing Programs" (Fox and Weisberg, draft) as a model, but noting that for the ordered logit and probit models, I have shown only the log-likelihood and not the gradient.
- Return a list with the maximum-likelihood estimates of the coefficients, including the intercepts or thresholds (negative of the intercepts); the covariance matrix of the coefficients (obtained from the inverse-Hessian returned by `optim()`), the residual deviance for the model (i.e., minus twice the maximized log-likelihood), and an indication of whether or not the computation converged.
- The ordered logit and probit models may be fit by the `polr()` function in the **MASS** package (one of the standard R packages). You can use `polr()` to verify that your function works properly. To test your program, you can use the `WVS` dataset in the **effects** package. For testing purposes, use a simple additive model rather than the model with interactions given in `?WVS`.

3. *General Cumulative Logit and Probit Models:* The ordered logit and probit models of the previous problem make the strong assumption that all $m-1$ cumulative probabilities can be modeled with the same regression coefficients, except for different intercepts. More general versions of these models permit different regression coefficients:

$$\Pr(Y_i > j) = \Lambda\left(\alpha_j + \beta_{1j}X_{i1} + \cdots + \beta_{kj}X_{ik}\right),\ j = 1, \ldots, m-1$$

or

$$\Pr(Y_i > j) = \Phi\left(\alpha_j + \beta_{1j}X_{i1} + \cdots + \beta_{kj}X_{ik}\right),\ \ j = 1, \ldots, m-1$$

Program one or the other (or both) of these models. For your example regression, use a likelihood-ratio test to compare the more general cumulative logit or probit model to the more restrictive ordered logit or probit model of the preceding problem. This test checks the assumption of equal slopes. The cumulative logit and probit models (along with the ordered logit and probit models) can be fit by the `vglm()` function in the **VGAM** package.

4. *Numerical Linear Algebra:* A matrix is said to be in *reduced row-echelon form* when it satisfies the following criteria:

   (a) All of its nonzero rows (if any) precede all of its zero rows (if any).
   (b) The first entry (from left to right) — called the *leading entry* — in each nonzero row is 1.
   (c) The leading entry in each nonzero row after the first is to the right of the leading entry in the previous row.
   (d) All other entries are 0 in a column containing a leading entry.

   A matrix can be put into reduced row echelon form by a sequence of *elementary row operations*, which are of three types:

   (a) Multiply each entry in a row by a nonzero constant.

(b) Add a multiple of one row to another, replacing the other row.

(c) Exchange two rows.

Gaussian elimination is a method for transforming a matrix to reduced row-echelon form by elementary row operations. Starting at the first row and first column of the matrix, and proceeding down and to the right:

(a) If there is a 0 in the current row and column (called the *pivot*), if possible exchange for a lower row to bring a nonzero element into the pivot position; if there is no nonzero pivot available, move to the right and repeat this step. If there are no nonzero elements anywhere to the right (and below), then stop.

(b) Divide the current row by the pivot, putting a 1 in the pivot position.

(c) Proceeding through the other rows of the matrix, multiply the pivot row by the element in the pivot column in another row, subtracting the result from the other row; this zeroes out the pivot column.

Consider the following example:

$$\begin{bmatrix} -2 & 0 & -1 & 2 \\ 4 & 0 & 1 & 0 \\ 6 & 0 & 1 & 2 \end{bmatrix}$$

Divide row 1 by -2:

$$\begin{bmatrix} 1 & 0 & 0.5 & -1 \\ 4 & 0 & 1 & 0 \\ 6 & 0 & 1 & 2 \end{bmatrix}$$

Subtract 4 × row 1 from row 2:

$$\begin{bmatrix} 1 & 0 & 0.5 & -1 \\ 0 & 0 & -1 & 4 \\ 6 & 0 & 1 & 2 \end{bmatrix}$$

Subtract 6 × row 1 from row 3:

$$\begin{bmatrix} 1 & 0 & 0.5 & -1 \\ 0 & 0 & -1 & 4 \\ 0 & 0 & -2 & 8 \end{bmatrix}$$

Multiply row 2 by -1:

$$\begin{bmatrix} 1 & 0 & 0.5 & -1 \\ 0 & 0 & 1 & -4 \\ 0 & 0 & -2 & 8 \end{bmatrix}$$

Subtract 0.5 × row 2 from row 1:

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -4 \\ 0 & 0 & -2 & 8 \end{bmatrix}$$

Add 2 × row 2 to row 3:

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The matrix is now in reduced row-echelon form. The rank of a matrix is the number of nonzero rows in its reduced row-echelon form, and so the matrix in this example is of rank 2.

Problem: Write an R function to calculate the reduced row-echelon form of a matrix by elimination.

Programming hints:

- When you do "floating-point" arithmetic on a computer, there are almost always rounding errors. One consequence is that you cannot rely on a number being *exactly* equal to a value such as 0. When you test that an element, say x, is 0, therefore, you should do so within a tolerance — e.g., $|x| < 1 \times 10^{-6}$.
- The computations tend to be more accurate if the absolute values of the pivots are as large as possible. Consequently, you can exchange a row for a lower one to get a larger pivot even if the element in the pivot position is nonzero.

5. *A less difficult problem:* Write a function to compute *running medians*. Running medians are a simple smoothing method usually applied to time-series. For example, for the numbers 7, 5, 2, 8, 5, 5, 9, 4, 7, 8, the running medians of length 3 are 5, 5, 5, 5, 5, 5, 7, 7. The first running median is the median of the three numbers 7, 5, and 2; the second running median is the median of 5, 2, and 8; and so on. Your function should take two arguments: the data (say, `x`), and the number of observations for each median (say, `length`). Notice that there are fewer running medians than observations. How many fewer?

6. *Simulation:* Develop a simulation illustrating the central limit theorem for the mean: Almost regardless of the population distribution of X, the mean $\overline{X}$ of repeated samples of size n drawn from the population is approximately normally distributed, with mean $E(\overline{X}) = E(X) = \mu$, and variance $V(\overline{X}) = V(X)/n = \sigma^2/n$, and with the approximation improving as the sample size grows. Sample from a highly skewed distribution, such as the exponential distribution with a small "rate" parameter λ (e.g., $\lambda = 1$); use several different sample sizes, such as 1, 2, 5, 25, and 100, and draw many samples of each size, comparing the observed distribution of sample means with the approximating normal distribution. Exponential random variables may be generated in R using the `rexp()` function. Note that the mean of an exponential random variable is $1/\lambda$ and its variance is $1/\lambda^2$.